

# System Automation mit Puppet und Foreman

Toni Schmidbauer

30. April 2014

# whoami

- ▶ SysAdmin@s-itsolutions
- ▶ toni@stderr.at
- ▶ <http://stderr.at>
- ▶ <http://github.com/tosmi>
- ▶ stderr@jabber.org

# Agenda

- ▶ Kurze Umfrage
- ▶ Was ist Puppet?
- ▶ Was ist Foreman?
- ▶ Puppet@s-iTSolutions
- ▶ Was haben wir geplant?

# Umfrage

# Was ist Puppet?

- ▶ Declarative programming: telling the machine what you would like to happen, and let the computer figure out how to do it.
- ▶ Imperative programming: telling the machine how to do something

```
1  class linuxwochen2014 (
2      $ensure = present
3  ) {
4      user { 'toni':
5          ensure => $ensure,
6          uid    => 4711,
7          gid    => 100,
8      }
9
10     package { 'emacs-nox':
11         ensure => installed
12     } ->
13     package { 'vim-enhanced':
14         ensure => absent,
15     }
}
```

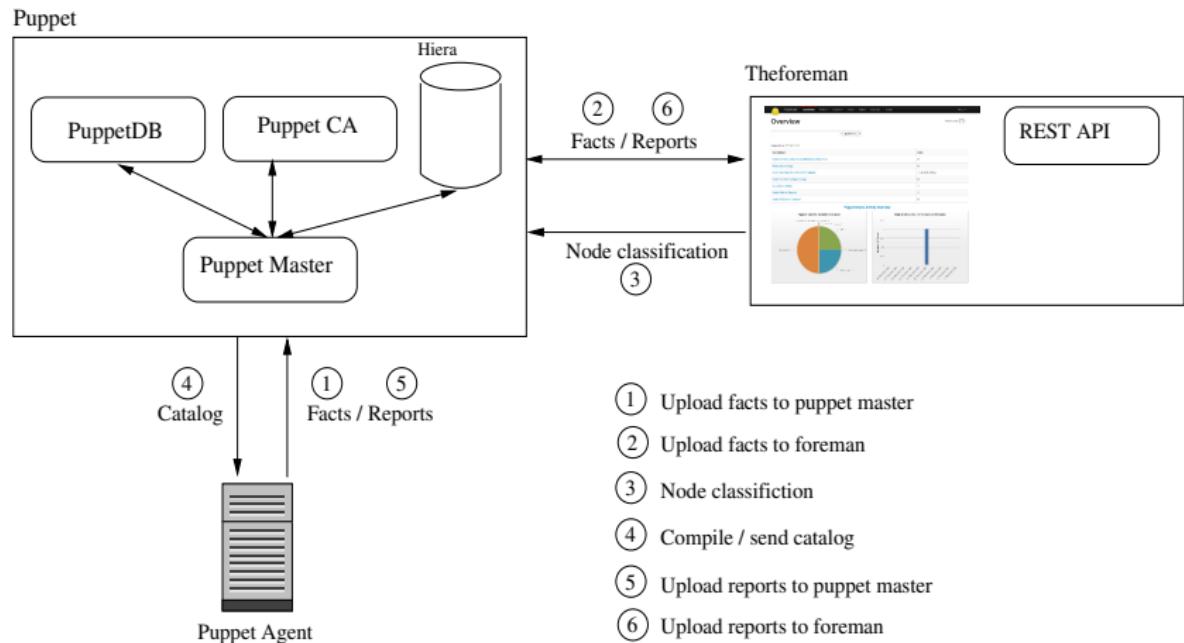
# Zuordnung von Klassen

- ▶ über manifests/site.pp

```
2     node /^(foo|bar)\.linuxwochen\.at$/ {  
3         class { 'linuxwochen2014':  
4             ensure => absent  
5         }  
6     }
```

- ▶ über einen External Node Classifier (Foreman)
- ▶ über Hiera (hiera\_include('classes',[““]))

# Puppet run



# Was ist Foreman?

 FOREMAN

Dashboard Hosts Reports Facts Audits Statistics Trends More ▾

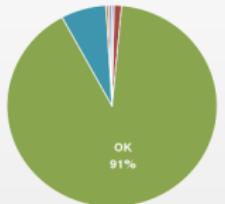
## Overview

Filter ...   ▾

Host Configuration Status

Hosts that had performed modifications without error	3
Hosts in error state	11
Good host reports in the last 35 minutes	875
Hosts that had pending changes	0
Out of sync Hosts	67
Hosts with no reports	4
Hosts with alerts disabled	5

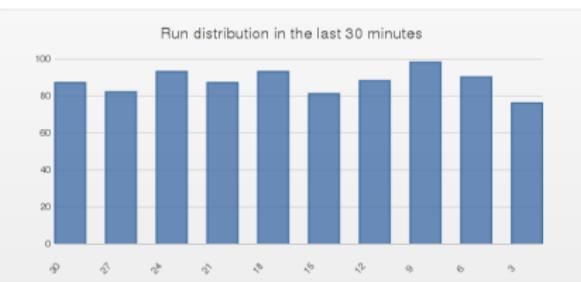
Total Hosts: 965



Latest Events

Host	A	R	F	FR	S	P
appplusu2.l...	0	0	0	0	8	0
appqsys2.se...	0	0	0	0	8	0
appplusu2.i...	0	0	0	0	8	0
appqsys2.se...	0	0	0	0	8	0
appplusu2.l...	0	0	0	0	8	0
appqsys2.se...	0	0	0	0	8	0

Run distribution in the last 30 minutes





# Und jetzt?



- ▶ Wie soll eine Entwicklungsumgebung aussehen?
- ▶ Wie testen wir den Puppet Code?
- ▶ Wie verwalten wir unseren Puppet Code?
- ▶ Wie soll unsere Puppet Umgebung aussehen?
- ▶ Wie erfolgt das Deployment des Codes?
- ▶ Wie verwalten wir Module von PuppetForge?

# Wie soll eine Entwicklungsumgebung aussehen?

# Vagrant

- ▶ <http://vagrantup.com>
- ▶ Ermöglicht virtuelle Entwicklungsumgebungen
- ▶ Vagrant Box ist ein vorkonfiguriertes Image
- ▶ Default VirtualBox andere Provider via Plugins (VMWare, KVM)

# Demo

# Wie testen wir den Puppet Code?

# rspec-puppet

- ▶ Ruby RSpec Tests für Puppet
- ▶ Jedes Module muss RSpec Tests mitbringen

```
1 require 'spec_helper'
2 describe 'linuxwochen2014' do
3   let :facts { { :osfamily => 'RedHat' } }
4
5   context 'ensure is set to absent' do
6     let :params { { :ensure => 'absent' } }
7
8     it do
9       should contain_user('toni').with({
10         'ensure' => 'absent',
11         'uid'    => '4711',
12         'gid'    => '100',
13       })
14     end
15
16     it { should contain_package('emacs-nox').with_ensure('installed') }
17     it { should contain_package('vim-enhanced').with_ensure('absent') }
18     it { should contain_package('emacs-nox').that_comes_before('Package[vim-enhanced]') }
19   end
20 end
```

# Demo

# Wie verwalten wir unseren Puppet Code?

# GIT

- ▶ Ein zentrales GIT Repository
- ▶ 3 Hauptbranches
  - ▶ Master
  - ▶ Testing
  - ▶ Production
- ▶ Feature Branches für neue Module
- ▶ Berechtigungssystem mit Gitolite

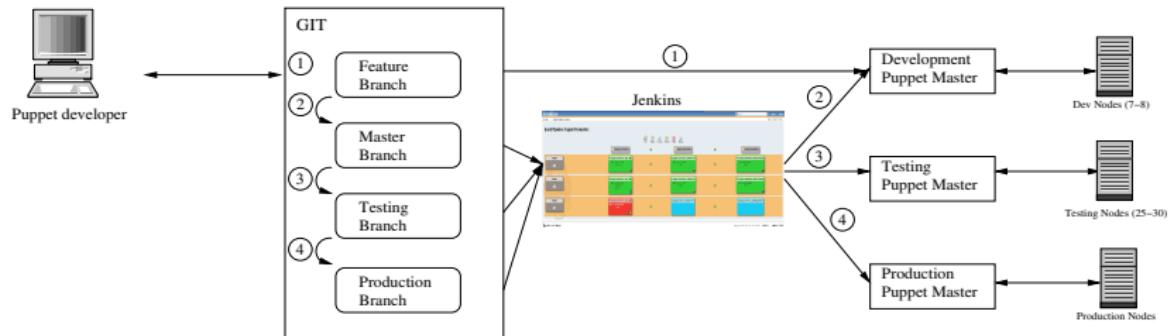
Wie soll unsere Puppet Umgebung aussehen?

Wie erfolgt das Deployment des Codes?

# Umgebungen

- ▶ Development (Master)
  - ▶ Test auf 7 Entwicklungsserver
  - ▶ Push auf Remote Master Branch löst Deployment aus
  - ▶ RedHat 5,6 / Solaris sparc,i386 10,11 / AIX
- ▶ Testing
  - ▶ ca. 30 "Produktions" Server
  - ▶ Annotated Tags werden automatisch deployed (test\_\*)
- ▶ Production
  - ▶ ca 1000 Hosts
  - ▶ Annotated Tags werden automatisch deployed (prod\_\*)

# Puppet Umgebung und Deployment



- (1) Features Branches get automatically created on Puppet Master (Dynamic Environments)
- (2) Master Branch gets deployed on commit via Jenkins
- (3) Testing Branch gets deployed via GIT tag (test\_\*)  
a normal commit to the Testing branch only runs tests
- (4) Production Branch gets deployed via GIT tag (prod\_\*)  
a normal commit to the Production branch only runs tests

It's all the same for Hiera yaml files, except dynamic environments!

# Deployment

Jenkins | Puppet Production Pipeline | [Search](#) | [Help](#) | [Log out](#) | [ENABLE AUTO REFRESH](#)

## Build Pipeline: Puppet Production

Perf History Configure Add Step Delete Merge

Pipeline	Step	Status	Duration
#3	#1 puppet_production_update_bit	Failed	~ 1 min 34 sec
#3	#2 puppet_production_deployment	Success	~ 1 sec
#3	#2 puppet_production_update_bit	Success	~ 1 sec
#2	#2 puppet_production_update_bit	Failed	~ 1 min 10 sec
#2	#1 puppet_production_deployment	Success	~ 1 sec
#2	#1 puppet_production_update_bit	Success	~ 1 sec
#1	#1 puppet_production_update_bit	Failed	~ 1 min 10 sec
#1	puppet_production_deployment	Success	~ 1 sec
#1	puppet_production_update_bit	Success	~ 1 sec

Help on this page | [Page generated: Apr 5, 2014 11:46:00 AM](#) | [REST API](#) | Jenkins ver. 1.532.2

# Monitoring

Monitor					
#55	puppet_development_deployment	1s	#5	puppet_development_hieradata	0s
#3	puppet_development_hieradata_check	0s	#2797	puppet_development_rspec_lint	1m 35s
#51	puppet_development_update_foreman	57s	#2	puppet_production_deployment	5s
#1	puppet_production_hieradata	1s	#5	puppet_production_hieradata_check	0s
#3	puppet_production_rspec_lint	1m 5s	#3	puppet_production_rspec_lint_nodeploy	1m 35s
#2	puppet_production_update_foreman	17s	#13	puppet_testing_deployment	2s
#4	puppet_testing_hieradata	1s	#3	puppet_testing_hieradata_check	0s
#2769	puppet_testing_rspec_lint	1m 35s	#8	puppet_testing_rspec_lint_nodeploy	1m 36s
#5	puppet_testing_update_foreman				

# Wie verwalten wir Module von PuppetForge?

# Puppetforge Module

- ▶ Eigenes GIT Repository (`puppetforge.git`)
- ▶ Download der Module in der Entwicklungsumgebung via  
`puppet module install ...`
- ▶ Staging GIT pull (bäh!)
- ▶ Dies ändert sich allerdings (dazu später)

- ▶ Wie soll eine Entwicklungsumgebung aussehen? *DONE*
- ▶ Wie testen wir den Puppet Code? *DONE*
- ▶ Wie verwalten wir unseren Puppet Code? *DONE*
- ▶ Wie soll unsere Puppet Umgebung aussehen? *DONE*
- ▶ Wie erfolgt das Deployment des Codes? *DONE*
- ▶ Wie verwalten wir Module von PuppetForge? *DONE*



# Probleme, Probleme, Probleme...

- ▶ Ein GIT Repo funktioniert nicht bei Änderungen von Upstream Modulen
- ▶ Andere Abteilungen sollen ihre Module unabhängig testen
- ▶ Unitests sagen noch nichts aus wie sich der Code am Live-System verhält
- ▶ Wir sollten eigentlich das Zusammenspiel aller Module testen (Forge und eigene)

# Was haben wir geplant?

- ▶ r10k für Deployment  
(<https://github.com/adrienthebo/r10k>)
- ▶ Ein Repository pro Module
- ▶ Nur interne Module bleiben im Hauptrepo
- ▶ Acceptance Tests mit Beaker

# Links und Bücher

- ▶ <http://github.com/tosmi/linuxwochen2014>
- ▶ <http://github.com/tosmi/puppet-devel>
- ▶ Puppet Learning VM: <http://puppetlabs.com/download-learning-vm>
- ▶ Foreman: <http://theforeman.org>
- ▶ Vagrant: <http://vagrantup.com>
- ▶ rspec-puppet: <http://rspec-puppet.com/>
- ▶ puppet-lint: <http://puppet-lint.com/>
- ▶ Gitolite: <http://gitolite.com/>
- ▶ r10k: <https://github.com/adrienthebo/r10k>
- ▶ Roles and Profiles: <http://www.craigdunn.org/2012/05/239/>
- ▶ Dynamische Puppet Umgebungen:  
<http://puppetlabs.com/blog/git-workflow-and-puppet-environments>
- ▶ puppet-sync: <https://github.com/pdxcat/puppet-sync>
- ▶ Instant Puppet 3 Starter
- ▶ Pro Puppet 2nd
- ▶ The RSpec Book
- ▶ Pulling Strings with Puppet
- ▶ Continuous Delivery

Danke für die Aufmerksamkeit